OULUN SEUDUN
AMMATTIKORKEAKOULU

Kai Heikka

MOBIILISOVELLUKSEN TOTEUTUS WEB-TEKNIIKOILLA PHONEGAP-KEHYKSELLE

MOBIILISOVELLUKSEN TOTEUTUS WEB-TEKNIIKOILLA PHONEGAP-KEHYKSELLE

Kai Heikka
Opinnäytetyö
Kevät 2012
Tietojenkäsittelyn koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä: Kai Heikka
Opinnäytetyön nimi: Mobiilisovelluksen toteuttaminen web-tekniikoilla PhoneGap-kehykselle
Työn ohjaaja: Liisa Auer
Työn valmistumislukukausi ja – vuosi: Kevät 2012 Sivumäärä: 36

Opinnäytetyön tavoitteena oli tutkia tablet-laitteiden sopivuutta kartoittaessa rannikoiden tilaa öljyvahinkojen sattuessa, jotta tilanteesta saadaan mahdollisimman ajantasainen ja oikea tieto tilanteen johtamiseen. Ensisijaisesti tutkimuksen kohteena oleva tablet-laite oli iPad 2, mutta tutkimuksen aikana todettiin myös muidenkin tablet-laitteiden käytön olevan mahdollista käyttämällä esimerkiksi web-tekniikoihin perustuvaa PhoneGap-kehystä.

PhoneGap-kehys oli sopiva opinnäytetyön tarpeisiin, koska se sisälsi tarvittavat ominaisuudet kuten kameran ja GPS-tietojen käyttämisen, joten se valittiin mobiilisovelluksen kehitykseen mukaan. PhoneGap myös sisältää osittain ilmaisen pilvipalvelun, jossa voi kääntää ohjelmakoodin suoraan monelle eri alustalle asennettavaksi asennuspaketiksi. Tätä ominaisuutta käytettiin paljon hyväksi mobiilisovelluksen kehityksen yhteydessä.

Opinnäytetyön tietosisällössä keskitytään mobiilisovelluksen luontiin sekä PhoneGap-kehyksen käyttöön. Lisäksi kerrotaan myös hieman ASP.NET MVC 3 palvelinsovelluksen toteuttamisesta. Mobiilisovelluksessa käytetään kahta uutta web-tekniikkaa, nämä ovat localStorage ja Web SQL Database. Kummatkin tekniikat ovat käytössä tiedontallennusta varten. Käytännössä localStorageen tallennetaan web-sivut sekä JavaScriptit. Web SQL Databaseen tallennetaan kuvat ja muut vahinkojen kartoitustiedot.

Opinnäytetyön tuloksena valmistui prototyyppi, jolla käy selväksi se, että tablet-laitteet soveltuvat hyvin vahinkojen kartoittamistietojen keräykseen. Niille saa tehtyä helppokäyttöisiä sovelluksia, joiden opetteluun ei mene paljoa aikaa. Suurimmat ongelmat olivat akunkesto sekä laitteen näytön sotkeutuminen, mutta näihin löytyy ratkaisuja kuten suojakalvoja sekä kenttälatureita.

Asiasanat: mobiili, phonegap, tablet, javascript, asp.net, html5

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Kai Heikka
Title of thesis: Implementation of mobile application for PhoneGap framework using web techniques
Supervisor: Liisa Auer
Term and year when the thesis was submitted: Spring 2012 Number of pages: 36

The primary objective of this thesis was to study the suitability of tablet devices in surveying coastal areas after oil spills to get up-to-date and correct information for leading the situation. The main device used in the study was iPad 2 but it was found that other devices could also be used for the same purpose. This was made possible by using a platform such as PhoneGap.

The PhoneGap was chosen as the platform for the development of the mobile application because it had all the necessary features, such as support for camera and GPS-data. It also includes a partially free to use cloud service that makes it possible to compile the application for multiple mobile platforms as a stand-alone installer that is distributable via different web stores. This cloud service was used often while developing the mobile application.

Information content of the thesis focuses on developing the mobile application by using PhoneGap framework in addition to it also overviews development of ASP.NET MVC 3 service application. Two new web techniques are used in the mobile application. These are localStorage and Web SQL Database. Both techniques are used for saving data. LocalStorage contains web pages and JavaScript files. Web SQL Database contains captured photos and survey information.

Result of the study was a working prototype which included part of the system functionality. It was enough to make clear that tablet devices fit well for surveying coastal areas after oils spills. And they allow for easy-to-use applications which do not take much time to learn. Most apparent issues were battery duration and tablet display getting dirty but there are solutions for these issues like tablet display screen protectors and portable chargers.

Keywords: mobile, phonegap, tablet, javascript, asp.net, html5

SISÄLLYS

1 JOHDANTO

Tässä opinnäytetyössä esitellään PhoneGap-kehystä hyödyntävän mobiilisovelluksen toteuttaminen. PhoneGap on mobiilialusta, jossa ohjelmointi tapahtuu JavaScriptiä sekä HTML:ää käyttäen. Alusta tukee kaikkia suosittuja mobiilikäyttöjärjestelmiä, joten saman sovelluksen saa helposti usealle eri mobiilikäyttöjärjestelmälle.

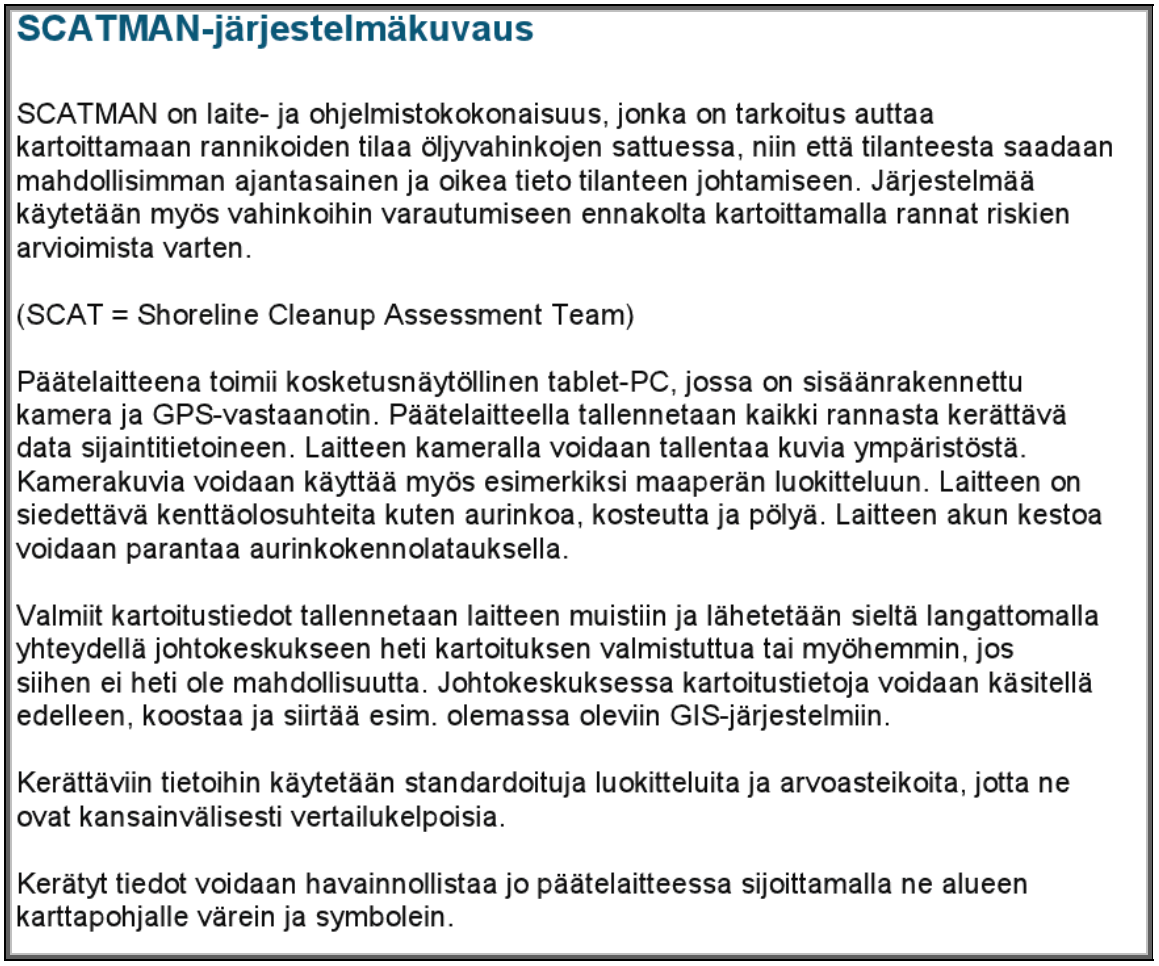
Mobiilisovellukseen liittyen toteutetaan myös palvelusovellus ASP.NET:llä, johon mobiilisovelluksella kerättyjä tietoja tallennetaan ja josta mobiilisovellukselle haetaan tarvittavaa tietoa sekä päivityksiä.

Mobiilisovelluksen käyttötarkoitus on auttaa kartoittamaan rannikoiden tilaa öljyvahinkojen sattuessa, niin että tilanteesta saadaan mahdollisimman ajantasainen ja oikea tieto tilanteen johtamiseen. Mobiilisovelluksella tallennetaan kaikki rannasta kerättävä data sijaintitietoineen. Laitteen kameralla voidaan tallentaa kuvia ympäristöstä. Kamerakuvia voidaan myös käyttää esimerkiksi maaperän luokitteluun.

Mobiilisovelluksella kerätyt kartoitustiedot tallennetaan laitteen muistiin ja lähetetään sieltä Internetin yli johtokeskukseen palvelusovellukselle heti kartoituksen valmistuttua tai myöhemmin, jos siihen ei heti ole mahdollisuutta. Johtokeskuksessa palvelusovelluksella kartoitustietoja voidaan käsitellä edelleen, koostaa ja siirtää esimerkiksi olemassa oleviin Geographical Information Systems -järjestelmiin.

Opinnäytetyössä tehty mobiilisovellus on prototyyppi SCATMAN-järjestelmästä, jonka järjestelmäkuvaus on nähtävillä kuviossa 1. SCATMAN-järjestelmän laajuus ylittää käytössä olevan ajan opinnäytetyöhön, jonka takia järjestelmästä luodaan prototyyppi. Toteutettava prototyyppi sisältää perustoiminnot joiden avulla voidaan tutkia tablet-laitteiden sopivuutta öljyvahinkojen kartoittamiseen, joten se ei sisällä kaiken tarvittavan tiedon keräystä ja lähettämistä johtokeskukseen palvelusovellukselle. Erikseen johtokeskusta varten toteutettava palvelusovellus on myös prototyyppi,

joka sisältää tiedon tallennus toiminnot sekä toiminnon tablet-laitteille luodun sovelluksen päivittämiseen.



KUVIO 1. SCATMAN-järjestelmäkuvaus.

Opinnäytetyön toimeksiantajana on Siperia Systems Oy, joka on tietojärjestelmien asiantuntijayritys. Sen päätuotteena on Palvelunohjaus-ohjelmisto hyvinvointipalveluiden suunnitteluun, ohjaukseen ja seurantaan. Lisäksi se tarjoaa ohjelmistokehitys-, projektinjohto- ja konsultointipalveluita muillekin toimialoille.

2 KÄYTETYT TEKNIIKAT

Käytetyissä tekniikoissa on selitetty perusasiat päätekniikoista, joita opinnäytetyössä käytetään. Suuriosa selitetyistä tekniikoista on mobiilisovelluksessa käytettyjä, mutta selitettynä on myös mobiilisovelluksen ja palvelinsovelluksen väliset tiedonsiirtotekniikat.

2.1 JavaScript

JavaScript on Web-ympäristössä käytettävä komentosarjakieli. JavaScriptin tärkein ominaisuus on mahdollisuus lisätä Web-sivuille dynaamista toiminnallisuutta. Nykymuodoltaan JavaScript on dynaamisesti tyypitetty, tulkattava oliopohjainen kieli, jonka syntaksi perustuu löyhästi C-ohjelmointikieleen. Viimeisin kielen määrittely on JavaScript 1.8.5, joka pohjautuu EcmaScript-standardiin ECMA-262 Edition 3. Standardoitua JavaScriptiä kutsutaan nimellä ECMAScript. (Wikipedia 2011b, hakupäivä 11.12.2011.)

JavaScriptiä kehitetään standardin mukaiseksi yhdessä Netscapen ja ECMA-standardointiorganisaation kanssa. Useat selaimet tukevat standardin lisäksi myös lisätoiminnallisuuksia, kuten Mozilla-selainten E4x, joka on XML:n käsittelyyn erikoistunut kielten laajennus. (Wikipedia 2011b, hakupäivä 11.12.2011.)

Esimerkki minimaalisesta HTML 4.01 syntaksiin perustuvasta JavaScript esimerkistä Web-sivulla:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
	<head><title>simple page</title></head>
	<body>
	<script type="text/javascript">
		document.write('Hello World!');
	</script>
	<noscript>
		<p>Your browser either does not support JavaScript, or has JavaScript turned off</p>
	</noscript>
	</body>
</html>
```

Esimerkissä luodaan HTML 4.01 standardiin perustuva Web-sivu sekä lisätään siihen JavaScript, joka tulostaa tekstin Hello World!, mutta jos JavaScriptiä ei ole selaimessa käytössä, niin Web-sivulle tulostetaankin teksti Your browser either does not support JavaScript, or has JavaScript turned off.

JavaScriptiä voidaan liittää Web-sivulle myös muilla tavoin. Yleinen tapa lisätä JavaScriptiä on käyttää Web-sivulle liitettyjä JavaScript tiedostoja. Tällöin tiedostot yleensä liitetään sivuston <head> osiossa. Seuraavat kaksi esimerkkiä ovat JavaScript-tiedostojen liittämiseksi Web-sivulle. Ensimmäinen tapa on nykyinen tapa liittää JavaScript-tiedostoja Web-sivuille. Toinen tapa on vanhentunut tapa liittää JavaScript tiedostoja Web-sivuille.

Nykyinen tapa:

```
<script language="javascript" type="application/javascript" src="script/jquery.js"></script>
```

Vanhentunut tapa:

```
<script language="javascript" type="text/javascript" src="script/jquery.js"></script>
```

Tapojen välillä olevana erona on tyyppi minä skript lisätään. Nykyisessä tavassa se on type="application/javacript" ja vanhentuneessa type="text/javascript". (RFC 4329 2012, hakupäivä 11.6.2012.)

JavaScriptiä voidaan lisätä Web-sivuille myös laittamalla se suoraan html elementtien tapahtumiin kuten onclick. Esimerkiksi link-elementin onclick tapahtumaan voidaan laittaa JavaScriptiä seuraavasti: Hei –popup. Voidaan myös käyttää omaa metodia suoraan linkissä: Oman metodin kutsuminen . JavaScriptiä voidaan käyttää myös useissa muissa eri elementtien eri tapahtumissa. (Korpela 2009, hakupäivä 11.12.2011.)

JavaScriptin ohjelmointiin on muutamia tunnettuja sovelluksia kuten Eclipse, Netbeans sekä Visual Studio. JavaScriptiä voidaan tosin kirjoittaa muun muassa pelkällä tekstieditorilla. JavaScriptin virheiden etsintä tapahtuu yleensä selaimilla ja niiden lisäosilla.

2.2 Web Storage

Web Storage on HTML5-standardiin perustuva tiedontallennusmenetelmä. Web Storage sisältää kaksi eri tapaa tallentaa tietoa. Nämä tavat ovat localStorage sekä sessionStorage. Nämä tavat eroavat toisistaan sillä, että sessionStorage*en* tallennetut tiedot pysyvät muistissa siihen saakka kun kyseisen sessionStorage*a* käyttävän Web-sivun ikkuna suljetaan. Joten tieto pysyy muistissa vaikka siirrytään samalla verkkotunnuksella olevalle eri sivuille. localStorage eroaa session-Storage*sta* siten, että siinä tieto pysyy localStoragessa tallessa vaikka selain suljetaan. Tieto pysyy tallessa niin kauan kunnes se nollataan sivuston kautta tai selaimen välimuisti tyhjennetään. Tietovaraston koko rajoitukseksi suositellaan viittä megatavua. (W3C 2011, hakupäivä 11.12.2011.)

Tuki Web Storagelle löytyy kaikista suosituimmista selaimista. Mutta vanhentuneet selaimet, kuten IE8 eivät tue Web Storagea. Myös osissa puhelimien selaimia on puutteita Web Storagen tuelle. (HTML5ROCKS, hakupäivä 11.12.2011.)

Web Storagen tuki voidaan tarkastaa seuraavalla tapaa:

```
if (window['localStorage']){
   //localStorage on tuettu
}
else {
   //localStorage ei ole tuettu
}
```

Web Storagea voidaan käyttää kahdella eri tavalla. Joko käyttäen metodeita arvojen lukemiseen, tai sitten lukemalla / tallentamalla suoraan muuttujanimien perusteella. Esimerkki arvojen tallentamisesta metodien avulla (W3C 2011, hakupäivä 11.12.2011.):

```
localStorage.setitem('muuttujanNimi','muuttujanArvo');
alert(localStorage.getitem('muuttujanNimi');
```

Edellä oleva näyttää popup-ikkunan, jossa lukee muuttujan *muuttujanNimi* arvo eli *muuttujan-Arvo*. Toinen tapa tehdä sama asia on:

```
localStorage.muuttujanNimi = 'muuttujanArvo';
alert(localStorage.muuttujanNimi);
```

2.3 Web SQL Database

Web SQL Database mahdollistaa tiedon tallentamisen verkkosivuilla JavaScriptillä SQL-kyselyiden avulla. Tietokanta pohjautuu SQLiteen ja on rajoitettu kooltaan. Perusrajoituksena tietokannan koolle käytetään viittä megatavua, mutta tarvittaessa selaimet osaavat kysyä käyttäjältä mahdollisesta tietokannan laajentamistarpeesta, joten tietokannat voivat olla sadankin megatavun kokoisia. (Wikipedia 2011a, hakupäivä 30.11.2011.)

Selaimet jotka tällähetkellä tukevat Web SQL Databasea ovat Google Chrome, Opera ja Safari. Tukea ei ole Mozilla Firefoxissa eikä Internet Explorerissa. Firefoxissa on kuitenkin tuki Indexed Database API:lle. (Wikipedia 2011a, hakupäivä 30.11.2011.)

Web SQL Database määrittelyn toteutti W3C, mutta tällä hetkellä sitä ei enää ylläpidetä, koska siihen ei löytynyt korvaavaa toteutusta, joka ei olisi käyttänyt SQLitea. (W3C 2010, hakupäivä 11.6.2012.)

2.4 PhoneGap

PhoneGap on HTML5-sovellusalusta, jolla voi toteuttaa Web-pohjaisia sovelluksia mobiililaitteisiin ja saada käyttöön mobiililaitteiden ominaisuuksia kuten kameran sekä GPS-paikannuksen. PhoneGapissä sovelluksien toteuttaminen tapahtuu HTML:n ja JavaScriptin avulla.

PhoneGap tukee useaa eri mobiilialustaa, joten ei ole tarvetta tehdä kokonaan eri sovellusta jokaiselle mobiilialustalle erikseen. On riittävää tehdä yksi PhoneGapiä hyödyntävä HTML5-pohjainen Web-sivusto, joka sitten voidaan ottaa käyttöön kaikilla tuetuilla alustoilla koodia muuntamatta. Tällä hetkellä tuettuja mobiilialustoja ovat iOS, Android, BlackBerry, Windows Phone, WebOS sekä Symbian. Eri toimintojen tuettavuus on tosin vaihtelevaa alustojen välillä. Esimerkiksi Symbian-käyttöjärjestelmässä on useita puutteita tuetuille ominaisuuksille.

Seuraavassa kuvassa (Kuvio 2) on listattuna tuetut toiminnot käyttöjärjestelmiin nähden.

KUVIO 2. Toimintojen tuki käyttöjärjestelmiin nähden.

PhoneGapin kotisivut sisältävät laajan dokumentaation toiminnoista sekä siitä, miten PhoneGap otetaan käyttöön eri alustoilla. PhoneGap myös tarjoaa osittain maksullisen palvelun siihen, että heidän kauttaan pilvipalvelussa voidaan käännättää omat PhoneGap-yhteensopivat sovellukset suoraan kaikille tuetuille PhoneGap-alustoille. Muuten esimerkiksi Applen iOS:lle tehdyt sovellukset vaativat Mac-laitteen kääntämistä varten. (PhoneGap 2011, hakupäivä 12.12.2011.)

2.5 ASP.NET MVC 3

MVC 3 -arkkitehtuuri on Microsoftin kehittämän ASP.NET MVC:n kolmas versio. MVC tulee sanoista model-view-controller eli malli-näkymä-käsittelijä. MVC-arkkitehtuuria käytetään etenkin graafisten käyttöliittymien suunnittelussa ja ohjelmoinnissa.

MVC:ssä ohjelma jaetaan kolmeen osaan: malliin, näkymään ja käsittelijään. Näistä malli kuvaa järjestelmän tiedon tallentamisen, ylläpidon ja käsittelyn. Näkymä määrittää käyttöliittymän ulkoasua ja tietojen näytön esityksen käyttöliittymässä. Käsittelijä toimii välittäjänä syötteelle muuttaen ne komennoiksi malliin tai näkymään. (Wikipedia 2011d, hakupäivä 12.12.2011.)

2.6 jQuery

jQuery on JavaScript-kirjasto, jota voidaan käyttää kaikilla yleisimmillä selaimilla. Sen tarkoitus on yksinkertaistaa ja helpottaa JavaScript ohjelmointia. jQueryllä kirjoitettu ohjelmakoodi toimii siis samalla tavalla kaikissa yleisemmissä selaimissa. Eri selaimille ei tarvitse kirjoittaa erilaista koodia niin usein kuin sitä normaalisti joutuisi kirjastoa käyttämättä. jQuery on tämän hetken suosituin JavaScript kirjasto, jota käytetään jopa 49 %:ssa suosituimpia Internet-sivustoja. (Wikipedia 2011e, hakupäivä 14.12.2011.)

Esimerkki jQueryn käytöstä:

```
$(document).ready(function(){
alert("Document is ready");
});
```

Yllä olevassa esimerkki koodissa luodaan jQueryllä metodi tapahtumaan, jossa Web-sivun elementit ovat valmiina. Metodissa näytetään popup-ikkuna, joka kuvaa tätä tapahtumaa.

2.7 jQuery Mobile

jQuery Mobile on jQueryä käyttävä kosketus-optimoitu JavaScript-kirjasto mobiililaitteille. Kirjastoa kehitetään samaan tapaan kuin jQueryä, eli erittäin selainyhteensopivaksi ja toimivaksi mahdollisimman monilla mobiililaitteilla. Kuviossa 3 nähdään mobiilikäyttöjärjestelmien tukitasot jQuery Mobilelle. Vihreä tarkoittaa parasta tukitasoa, keltainen keskitasoa ja punainen matalaa tasoa.

Platform	Version	Native	PhoneGap	Opera Mobile				Opera Mini		Fennec		Ozone	Netfront
			0.9	8.5	8.65	9.5	10.0	4.0	5.0	1.0	1.1*	0.9	4.0
iOS	v2.2.1	B	A										
	v3.1.3, v3.2	A	A						A				
	v4.0	A	A						A				
Symbian S60	v3.1, v3.2	C		C	C		B	C	B			C	C
	v5.0	A	A	C	C		A	C	A				
Symbian UIQ	v3.0, v3.1				C							C	
	v3.2					C						C	
Symbian Platform	v.3.0	A											
BlackBerry OS	v4.5	C						C	C				
	v4.6, v4.7	C	C					C	B				
	v5.0	B	A					C	A				
	v6.0	A	A						A				
Android	v1.5, v1.6	A	A										
	v2.1	A	A										
	v2.2	A	A				A*		C*		A*		
Windows Mobile	v6.1	C		C	C	C	B	C	B				C
	v6.5.1	C		C	C	A	A	C	A				
	v7.0	A					A	C	A				
webOS	**1.4.1**	A	A										
bada	**1.0**	A											
Maemo	**5.0**	B					B			C	B*		
MeeGo	1.1*	A*					A*				A*		

Key:

- **A - High Quality.** A browser that's capable of, at minimum, utilizing media queries (a requirement for jQuery Mobile). These browsers will be actively tested against, but may not receive the full capabilities of jQuery Mobile.
- **B - Medium Quality.** A capable browser that doesn't have enough market share to warrant day-to-day testing. Bug fixes will still be applied to help these browsers.
- **C - Low Quality.** A browser that is not capable of utilizing media queries. They will not be provided any jQuery Mobile scripting or CSS (falling back to plain HTML and simple CSS).
- * - Upcoming browser. This browser is not yet released but is in alpha/beta testing.

KUVIO 3. jQuery Mobilen mobiilikäyttöjärjestelmien tuki. (Wikipedia 2011f, hakupäivä 14.12.2011.)

jQuery Mobile myös parantaa tavallisien HTML-elementtien ulkoasua ja luettavuutta erityisesti mobiililaitteella. Esimerkiksi listoista saadaan enemmän interaktiivisia ja näyttävän näköisiä liittämällä pieniä lisäyksiä HTML-koodiin.

Esimerkiksi seuraavalla koodilla saadaan normaali `select`-elementistä tehtyä näyttävän näköinen jQuery Mobile `select`-elementti:

```
<select name="select-choice-15" id="select-choice-15" data-theme="b" data-overlay-theme="d" data-native-menu="false">
        <option value="t">Tall</option>
        <option value="g">Grande</option>
        <option value="v">Vente</option>
</select>
```

Koodista muodostuu elementti joka avatessa näyttää seuraavalta (Kuvio 4):

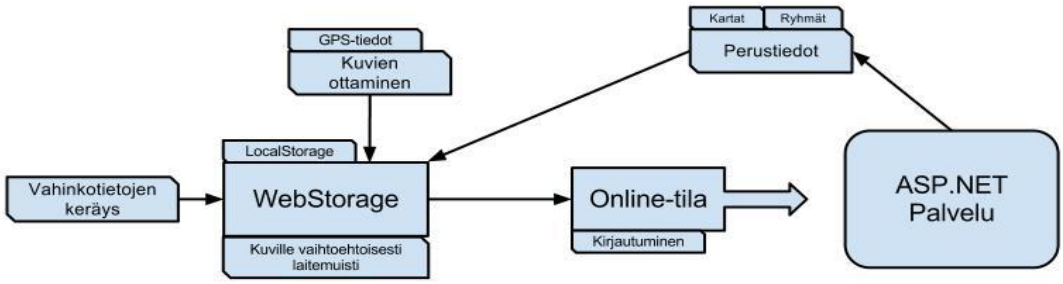
KUVIO 4. jQuery Mobilen avattu select-elementti (jQuery Mobile 2011, hakupäivä 14.12.2011).

3 ASP.NET PALVELU JA TIEDON SYNKRONOINTI

3.1 Tietojen synkronointi ja palvelun toiminta

Ennen kuin mobiilisovelluksen toteutus voitiin aloittaa kunnolla, niin sitä varten tarvittiin palvelusovellus. Palvelusovellusta käytetään tiedon hakuun, tallentamiseen ja datatiedostojen päivittämiseen mobiilisovelluksessa. Datatiedostoilla tarkoitetaan tiedostoja, joista mobiilisovellus koostuu. Mobiilisovellus voidaan päivittää automaattisesti hakemalla palvelusovelluksesta tuoreimmat datatiedostot.

Kuviossa 5 nähdään tiedon synkronointitapahtuma täydellisessä järjestelmässä. Täydellisellä järjestelmällä tarkoitetaan prototyypin jatkokehityksen jälkeistä järjestelmää. Prototyypissä palvelimelta testitarkoituksessa haetaan perustiedoista vain organisaatiot. Tämä riitti testaamaan tarvittavan toiminnallisuuden. Kuviossa 5 vasen puoli tarkoittaa mobiilisovellusta ja oikea puoli palvelusovellusta.



KUVIO 5. Tiedon synkronointi täydellisessä järjestelmässä.

Kun tiedot haetaan palvelusovelluksesta, niin siihen käytetään käsittelijässä seuraavaa tapaa.

```
public string data_getOptionalData(string d)
{
    Models.OptionalData optionalData = new Models.OptionalData();

            optionalData.organization = data_optional_getOrganizations();

            return Models.ScatSecurity.doB64Encode(JsonConvert.SerializeObject(optionalData));
}
```

Metodissa luodaan taulukko kaikista tietokannassa olevista organisaatioista optionalData-olioon ja se muutetaan JSON-muotoiseksi tekstiksi, jonka jälkeen se koodataan base64-muotoon. Muutokset auttavat tiedon siirrossa, sekä helpottavat muuttamista takaisin oliomuotoon JavaScript puolella mobiilisovelluksessa. OptionalData-malli suunniteltiin siten, että se voi sisältää paljon tietoa, organisaatiot ovat vain yksi esimerkki mahdollisesta tiedosta, mitä se voi sisältää.

JavaScriptissa voidaan muuttaa haettu optionalData helposti takaisin oliomuotoon seuraavasti.

```
var optionalData = null;
if (localStorage.getItem("optionalData") != null)
   optionalData = JSON.parse(localStorage.getItem("optionalData"));
```

Edellä olevassa koodissa tarkistetaan onko optionalData-tietoa tallennettuna, ja näin ollessa sen jälkeen se jäsennellään takaisin JavaScript-olioksi. JavaScript-oliota voidaan tämän jälkeen vaivattomasti käyttää joka puolella mobiilisovellusta esimerkiksi seuraavalla tavalla: optionalData.organization[0], tämä kutsu palauttaisi ensimmäisen organisaation nimen, joka ASP.NET palvelusovelluksesta haettiin.

Tietojen liikkumisessa mobiilisovelluksesta palvelinsovellukseen päin käytetään samantapaista oliopohjaista tapaa. Tiedot ovat tallessa survey-oliossa joka myös sisältää muita olioita kuten zone-, team- ja sample-oliot. Tiedon lähetyksen yhteydessä survey-olio ladataan tietokannasta ja se paloitellaan kuviksi ja tekstitiedoiksi. Tämä tehdään sen takia että kuvat voidaan lähettää yksi kerrallaan eikä mahdollista POST_MAX_SIZE ylitystä tapahdu. Aluksi lähetetään tekstitiedot, koska se tapahtuu nopeasti verrattuna kuvien lähetykseen ja sen perusteella näkee onko tietoja yritetty lähettää. Jos kuvien lähettäminen epäonnistuu jostain syystä, niin voidaan nähdä kumminkin se että jotain tietoja on yritetty lähettää ja tarvittaessa ottaa yhteyttä lähettäjään. Kuvia voi olla jopa sato-

ja ja niiden koot ovat noin sata kilotavua per kuva. Tietojen lähettämiseen palvelimelle käytetään seuraavaa koodia.

```
function sendGeneralSurveyInfo(UID, json)
{

  var parameters = "username=user&password=hashpass&UID=" + UID + "&json=" + Base64.encode(json);
  $.ajax({ type: "POST", data: parameters,
     url: 'http://' + serverAddress + '/Sync/SurveyUploader',
     complete:
          function (data, textStatus)
          {
             if (data.responseText == "SAVE_OK")
             {

                dbo.db.transaction(function (tx)
                {
                   tx.executeSql('UPDATE tblSurvey SET sent=? WHERE UID=?'
                   , [getCurrentTimeAndDate(), UID], dbo.onSuccess, dbo.onError);
                });

                sendSurveyImages(UID);
             }
             else
             {
                doAlert("Sending of survey failed. There was an error: " + data.responseText)
                sendingSurvey = false;
             }
          }
  });
  }
```

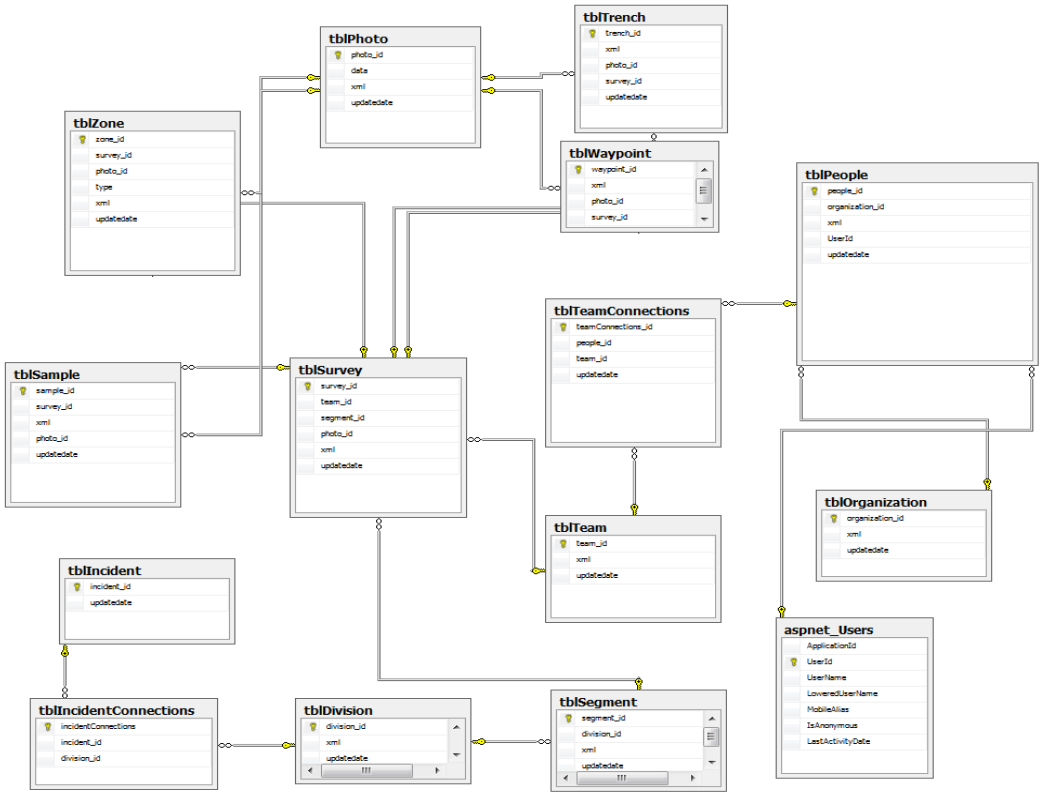
Edellä olevassa koodissa nähdään esimerkki tietokannan käytöstä sekä jQuery POST-tapahtumasta. Tapahtumassa päivitetään selvityksen tilaksi lähetetty, jos tapahtuma on toteutettu onnistuneesti. Samalla aloitetaan kuvien lähettäminen. Koodissa on käytössä esimerkkisalasanat, joita ei prototyypissä vielä tarkisteta.

Kuvien lähettäminen tapahtuu melkein vastaavasti, mutta niiden yhteydessä lisäksi päivitetään edistymispalkkia. Kuvien tallennustapahtuma palvelinsovelluksessa kuitenkin on monimutkaisempi, koska kuvat ovat tekstiä, ja ne täytyy muuttaa kuviksi. Tämän jälkeen kuviin myös lisätään EXIF-tiedot, jotka tarkoittavat kuvien sisältämää tietoa kuten GPS-tietoja.

3.2 Tietokanta

Tietokannan tutkimiseen ja suunnitteluun käytettiin paljon aikaa. Se tehtiin vastaamaan mahdollisimman hyvin oikeaa järjestelmää, joten se ei ole pelkästään prototyyppiä varten toteutettu tietokanta. Kuviossa 6 nähdään tietokanta kokonaisuudessaan. Prototyypissä tietokannasta on käytössä vain 3 taulua, jotka ovat aspnet_Users, tblPeople ja tblOrganization. Kahta jälkimmäistä taulua käytettiin testitarkoituksessa, ja aspnet_Users-taulua käytettiin kirjautumiseen, mutta siihen oli myös viittaus tblPeople-taulusta, jolla henkilötiedot voitiin liittää kirjautuneeseen käyttäjään.

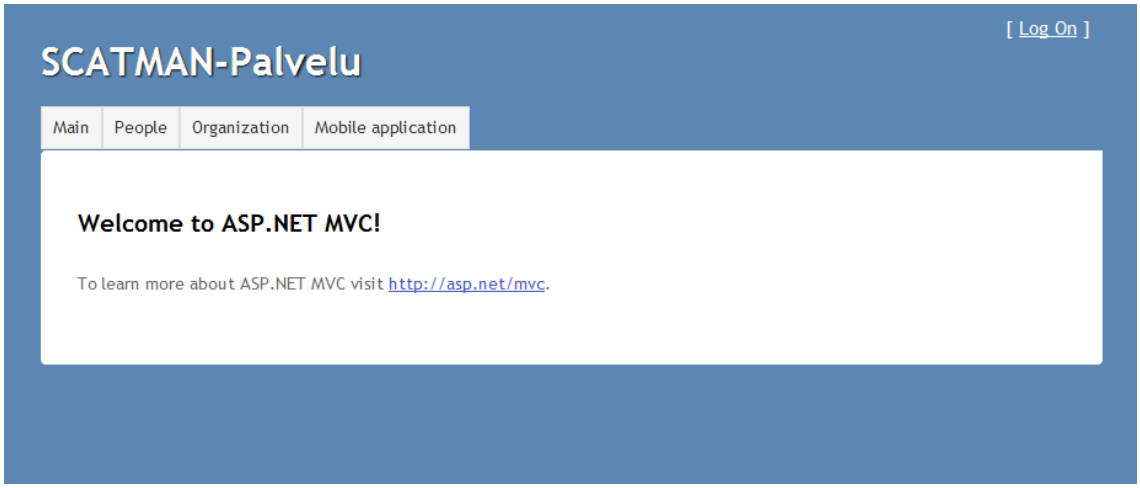
Tietokannan kaikki taulut muodostettiin lähes samoilla periaatteilla. Ne olivat kasvava perusavain, viittauksia varten olevat tärkeimmät arvot, viimeksi päivitetty -kenttä sekä xml-kenttä. Tietokannassa xml-kentän tärkeys on suuri, koska se sisältää kaikki arvot, joihin ei tarvitse suoraa viitata mistään taulusta, joten siellä arvoja voi olla tallessa useita kymmeniä per rivi. Xml-tiedostoihin on vaivaton jälkeenpäin lisätä uusia arvojen tallennuksia, joten se on toimiva ratkaisu jatkokehitystä ajatellen.



KUVIO 6. Tietokanta ja sen taulujen väliset yhteydet.

3.3 Käyttöliittymä

ASP.NET palvelinsovelluksessa käyttöliittymänä käytetään MVC3:n vakioulkoasua. Prototyypissä keskityttiin taustapalveluiden toteuttamiseen ja tietokannan käsittelyyn. Käyttöliittymä on keskeneräinen ja sisältää vain muutaman testaustoiminnon prototyyppiä varten. Kuviossa 7 nähdään otos käyttöliittymän etusivulta.



KUVIO 7. Palvelinsovelluksen käyttöliittymän aloitussivu.

4 MOBIILISOVELLUKSEN TOTEUTTAMINEN

4.1 Mobiilisovelluksen rakenne

Suunnitteluvaiheessa oli tiedossa, että iPad ei välttämättä ole laite millä mobiilisovellusta käytetään. Täten oli tärkeää, että sovellusta voidaan käyttää muillakin mobiililaitteella. Tätä varten tutkittiin eri mahdollisuuksia usealle alustalle kääntämistä varten.

Tutkimuksen jälkeen löytyi vakuuttava, nopeasti kehittyvä PhoneGap-kehys. Sillä kaikki prototyypille tarvittavat ominaisuudet saataisiin käyttöön suoraan JavaScriptin avulla, sekä sillä myös monelle eri alustalle kääntäminen oli mahdollista. Tutkimuksen jälkeen PhoneGap-kehys valittiin alustaksi ja testaus aloitettiin heti. Testauksen jälkeen suunniteltiin tapaa miten sovelluksen runko toteutetaan ja miten sovelluksen päivitys tapahtuu testauksen yhteydessä, ja sen valmistumisen jälkeen.

4.1.1 Mobiilisovelluksen päivitys

Mobiilisovelluksen päivittämisen suunnittelu ja toteutus oli suhteellisen suuri homma projektin aikana. Siitä haluttiin mobiilisovellukselle hyvä, toimiva sekä nopeasti tapahtuva prosessi, jotta pienienkin muutoksien testausta pystyttäisiin suorittamaan helposti sekä nopeasti suoraan mobiililaitteella. Ei ollut hyväksyttävää, että muutoksen testaus esimerkiksi kahdella eri tablet-laitteella vaatisi paria minuuttia enempää työtä. Tämän takia päädyttiin tapaan, jossa itse sovellus PhoneGap-kehyksen lisäksi sisältää vain muutaman tiedoston. Päätiedostona mobiilisovelluksessa on index.html, ja sen lisäksi on kuvatiedostoja sekä tyylitiedostoja, jotka ovat suhteellisen staattista sisältöä. Nämä tiedostot ovat siis PhoneGap-kehyksen sisällä ja näitä tiedostoja ei tarvitse siirtää itse laitteeseen kovinkaan usein testauksen yhteydessä, koska itse sovelluksen eri sivut ladataan sovelluksen ollessa online-tilassa ja tallennetaan localStorageen, josta niitä voidaan käyttää milloin vain. Mobiilisovelluksen käynnistyessä Internet-yhteyden ollessa saatavilla otetaan yhteys palvelimeen, josta ladataan kaikki sovelluksen käyttämä tieto, jos mobiilisovelluksen sen hetkinen tieto on vanhentunutta. Tietona ovat JavaScriptit sekä sovelluksen eri sivut. Nämä luokitellaan

sovelluksessa dynaamiseksi sisällöksi ja sovellus rakentuu näiden avulla. Kun halutaan tehdä mobiiilisovellukseen muutos, se tapahtuu muuttamalla jotain dynaamista tiedostoa, esimerkiksi JavaScript-tiedostoa. Muutoksen jälkeen mobiilisovelluksessa käytetään reset-toimintoa, joka lataa dynaamisen sisällön uudestaan palvelimelta, ja näin ollen muutos on mobiilisovelluksessa parin nopean toiminnon jälkeen. Tämä mahdollistaa nopeamman sovelluksen kehityksen ja testauksen.

Mobiilisovelluksen automaattinen päivitys on toteutettu siten, että palvelimella on kansio autoload, joka sisältää kaikki mobiilisovelluksen dynaamisesti ladattavat tiedostot. Prototyypissä kansiossa oli 35 tiedostoa, jotka aina ladattiin reset-toiminnon tai uuden version ollessa saatavilla. Mobiilisovelluksen päivitystä varten on versioasetus, jota päivittämällä mobiilisovellus lataa käynnistyksen yhteydessä uudet datatiedostot.

Sisältötiedostot ladataan jQueryn avulla yksi tiedosto kerrallaan ja ladattavien kohteiden lista muodostetaan tätä ennen palvelinsovelluksessa. Tiedostot tallennetaan localStorageen, josta ne avataan myöhemmin JavaScriptin avulla. Tähän käytetään jQuery.globalEval() ja jQuery.html().trigger("create") metodeita, josta enemmän seuraavassa osiossa 4.1.2.

4.1.2 Sisällönhallinta

Sisällönhallinnalla tarkoitetaan mobiilisovelluksen tiedostojen jaottelua, latausta ja miten niitä käytetään mobiilisovelluksessa. Sisällönhallinnassa käytetään hyödyksi JavaScriptiä sekä HTML5:n uutta ominaisuutta localStoragea. Lähes kaikki sivujen sisältö ovat tallennettuna suoraan localStorageen, joten sisältö voidaan ladata dynaamisesti sivulle, eikä sivunvaihtoja juuri tarvitse. Mobiilisovelluksessa on yksi tiedosto index.html, jossa määritetään osio, johon sivujen sisältö localStoragesta ladataan.

Sivujen vaihtoon mobiilisovelluksessa käytetään kahta eri tapaa. Toinen yleisempi tapa on, että käytetään metodia, joka dynaamisesti lataa sivun localStoragesta, ja toinen vähemmän käytetty tapa on, että ladataan sivu normaalilla sivunvaihdolla, mutta attribuuttien perusteella. Voidaan käyttää esimerkiksi linkkiä http://esimerkki.fi/index.html?page=help&sub=menu, joka lataisi help-sivujen

valikon. Tämä tapa on usein käytössä tavallisilla web-sivuilla lukuun ottamatta sitä, että tässä mobiilisovelluksessa sitä käytetään JavaScriptillä. Yleensä sisällönhallinta toimii palvelinpuolella esimerkiksi PHP:n avulla.

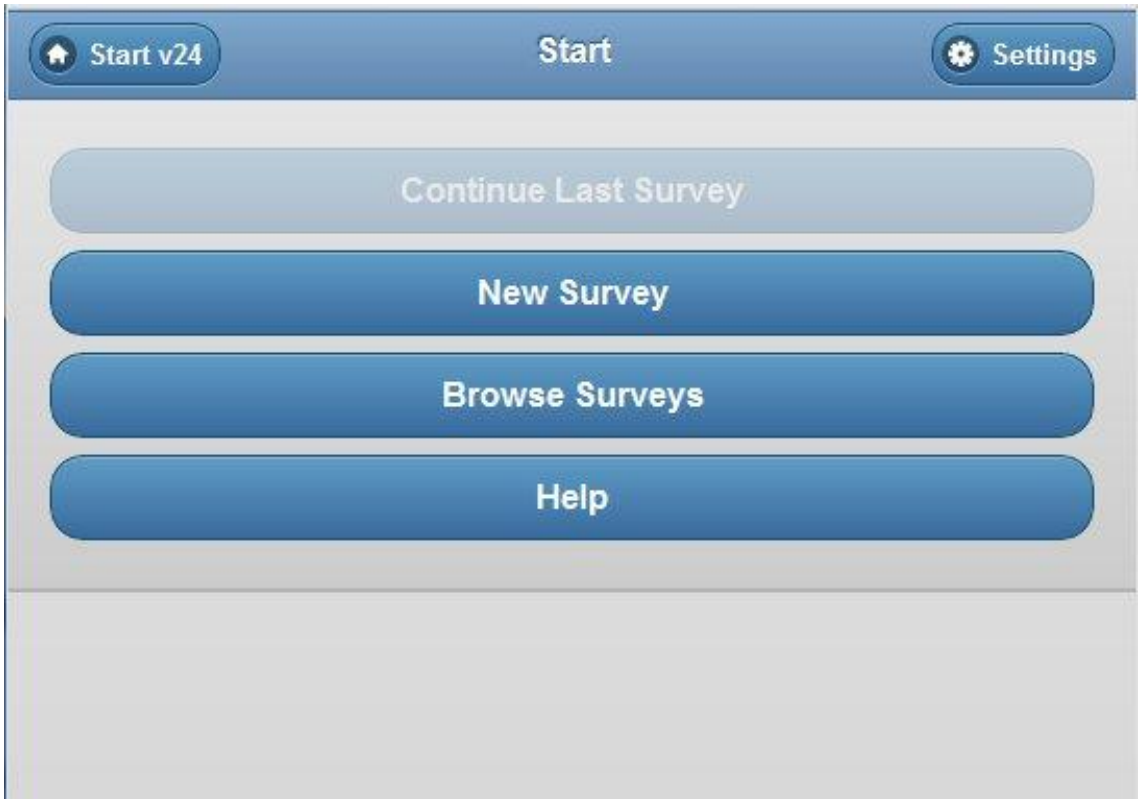
Uuden sisältösivun latauksessa mobiilisovelluksessa käytetään pääosin loadPage-metodia. Metodi tarkistaa sille annettujen parametrien avulla, minkä tyyppinen ladattava sisältösivu on. Prototyypissä sisältösivutyyppejä on kahdenlaisia, moniosaisia ja yksiosaisia. Moniosaisilla tarkoitetaan sisältösivua, jossa on navigointipalkki, ja yksiosasilla sisältösivua, jossa sitä ei ole. Näiden lataamista varten käytetään hieman erilaisia metodeita, mutta lopputulos on sama, sisältösivu latautuu dynaamisesti localStoragesta ilman sivun kokonaista latautumista.

Kun itse sisältö on ladattu, niin tarvitaan sivulle vielä tapahtumalaukaisimet. Niillä muun muassa tallennetaan tietokentät survey-olioon, rajoitetaan mitä tekstikenttiin voi kirjoittaa ja konvertoidaan sisältö vastaamaan valittua kielisyyttä.

4.2 Käyttöliittymä

Mobiilisovelluksen toteutus lähti esitestauksen jälkeen liikkeelle sopivan käyttöliittymän suunnittelusta. Käyttöliittymän suunnittelussa otettiin mallia muista mobiilisovelluksista kuten iPad-sovelluksista. Käyttöliittymän toteutusta varten oli jo ennalta sovittu JavaScript-kirjasto jQuery Mobile, jonka perusteella käyttöliittymää suunniteltiin.

Käyttöliittymän suunnittelun yhteydessä tapahtui myös sovelluksen toimintojen suunnittelu. Suunnittelun alkuvaiheessa ei ollut vielä tarkkaa tietoa siitä, minkälaista tietoa sovelluksella tulee kerätä. Joten suunnittelussa keskityttiin perustoimintoihin, jotka sovellukseen ainakin olivat tulossa.

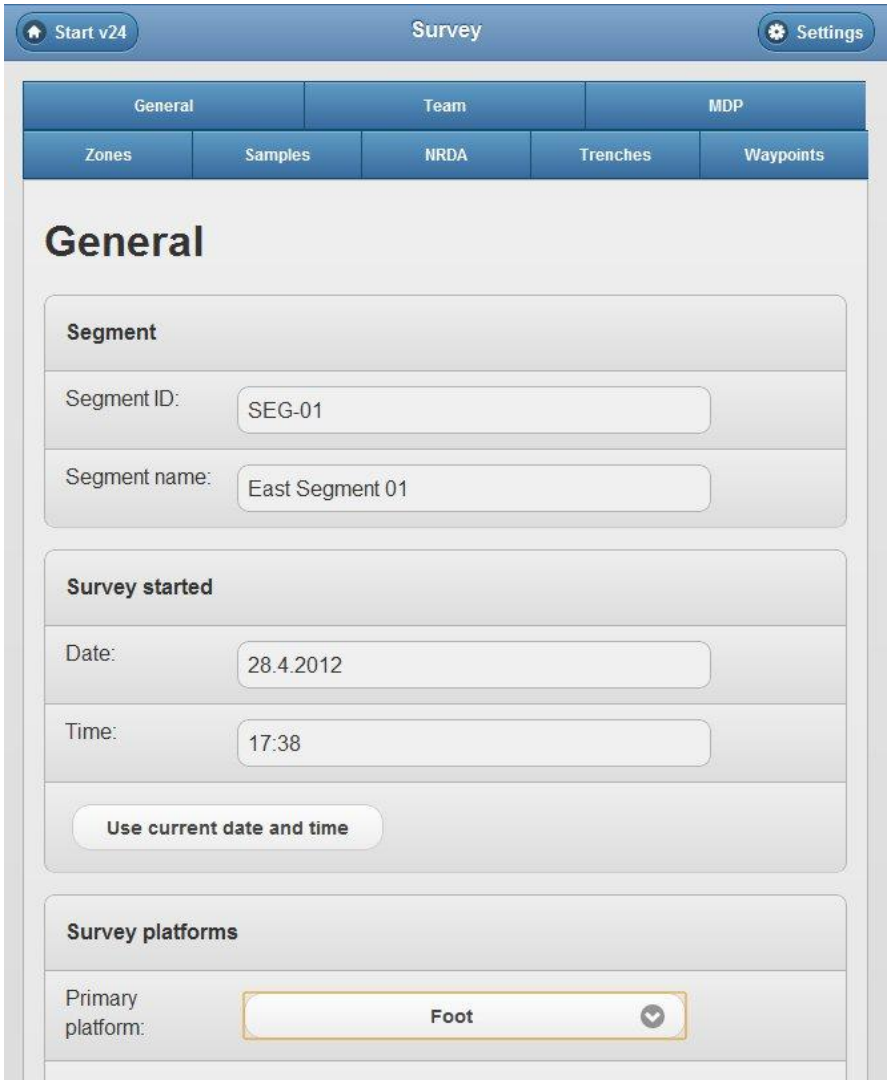


KUVIO 8. Käyttöliittymän alkuvalikko

Kuviossa 8 nähdään mobiilisovelluksen alkuvalikko, joka ensimmäisen toteutuksen jälkeen pysyi lähes samanlaisena prototyypin valmistumiseen asti. Käyttöliittymää ei erikseen suunniteltu millään sovelluksella, vaan se toteutettiin suoraan mobiililaitteisiin ja testattiin niillä toimivaksi. Toiminnot olivat puutteellisia vielä käyttöliittymän suunnitteluvaiheessa. Tuskin mitään toiminnallisuutta ei ollut toteutettu. Kuvio 8 onkin otettu valmiista prototyypistä, mutta tosin sisältää turhaa versiotietoa Start-napissa.

Tavoitteena käyttöliittymässä on mahdollisimman hyvä käytettävyys ja selvät valintavaihtoehdot. Myös on tärkeää, että käyttöliittymä näyttää aina mahdollisimman samantapaiselta ja esimerkiksi New Survey-painike on aina samassa kohdassa käyttöliittymässä. Tämän takia Continue Last Survey-painike näkyy käyttöliittymässä harmaana vaikka sitä ei ole mahdollista klikata kuvan ottohetkellä. Start- ja Settings-painikkeet näkyvät jokaisella sovelluksen sivulla, ja niiden toiminnallisuus on aina sama. Start-painikkeella pääsee sovelluksen aloitusnäkymään ja Settings-painikkeella voi esimerkiksi käynnistää sovelluksen uudelleen tai muuttaa palvelinta, johon sovellus synkronoi tiedot.

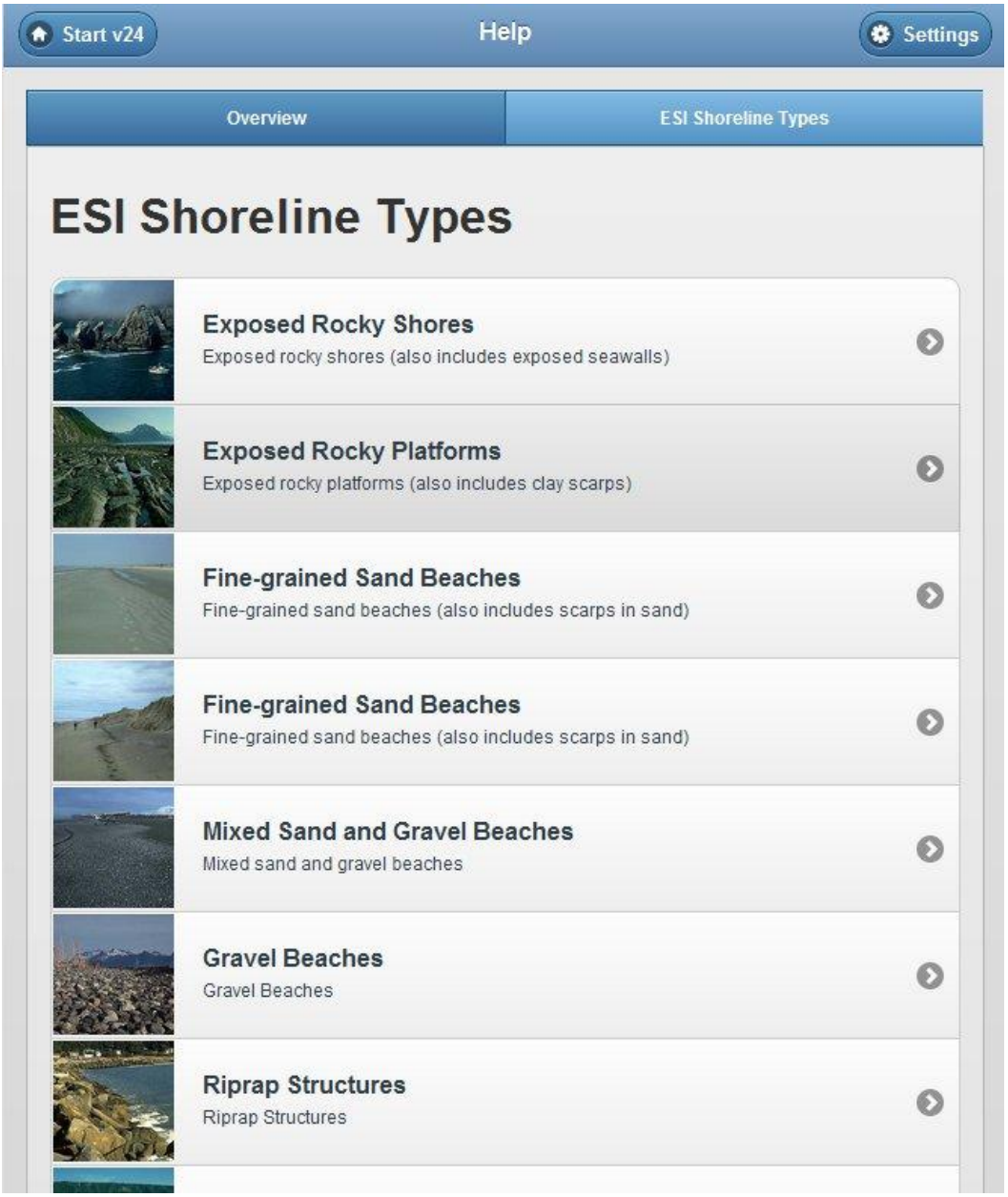
Seuraavassa kuviossa 9 nähdään sovelluksen General-välilehti, kun on aloitettu uusi selvitys (New Survey). Sisällön keruuseen otettiin käyttöön Windowsista tuttu välilehtiä sisältävä käyttöliittymä. Välilehdet mahdollistavat tutun käytettävyyden sekä selvän tietojen järjestelemisen. Tiedon keräyskentissä käytetään samanlaista toteutusta, joka katsottiin toimivaksi suunnittelun yhteydessä koko sovelluksessa. Käyttöliittymän muut sivut ovat siis samannäköisiä kuin tämä General-sivu, mutta ovat eri otsikoilla ja sisältöteksteillä.



KUVIO 9. Käyttöliittymän General-välilehti.

Prototyypin käyttöliittymän Help-sivut jäivät suhteellisen vähäisiksi. Ne sisältävät muutaman esimerkkidokumentin sekä ESI Shoreline Types-sivun (Kuvio 10), joka sisältää esimerkkejä rannikko-

tyypeistä. Rannikkotyyppiä klikkaamalla pääsee tarkempaan näkymään kyseisestä rannikkotyypistä. Prototyypissä tarkemmat näkymät sisältävät vain suurennetun kuvan.

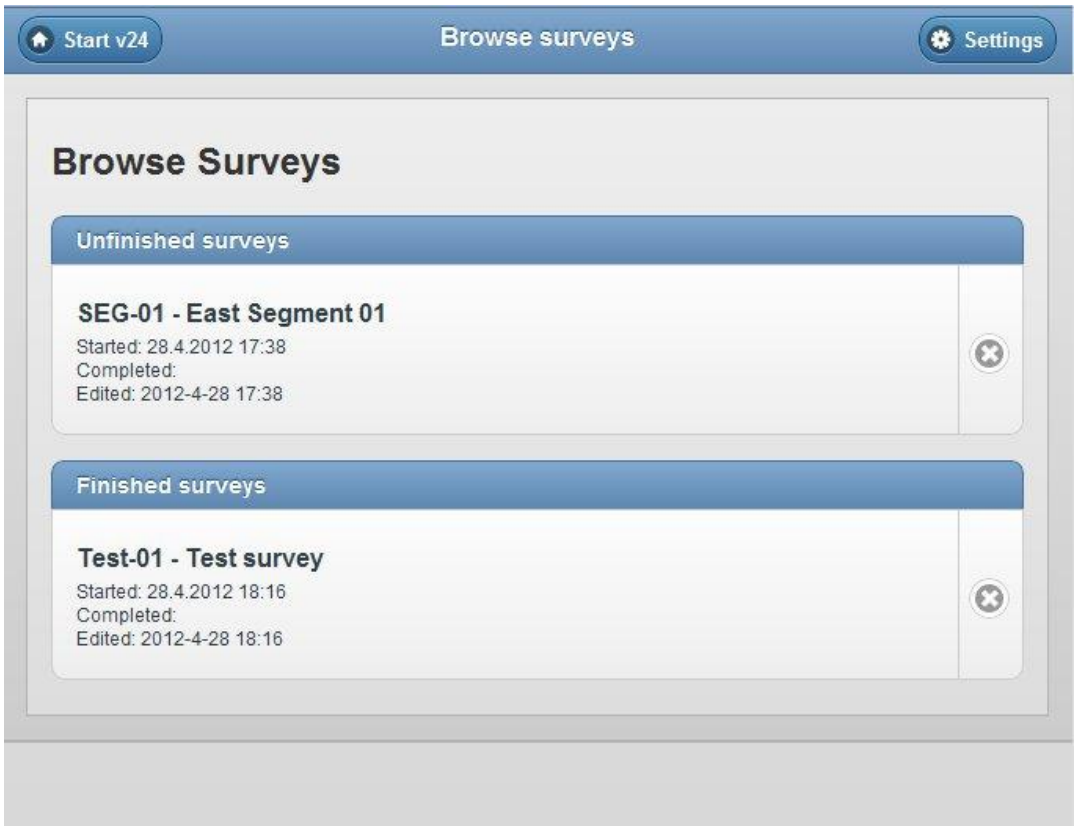


KUVIO 10. Help-sivun esimerkki katsaus.

4.3 Tietojen hallinta

Tietojen hallinnalla tarkoitetaan sitä, miten mobiilisovelluksen normaali käyttäjä voi hallinnoida kerättyä tietoa. Sovelluksessa käyttäjä voi selata tallennettuja selvityksiä (Browse Surveys) ja poistaa niitä tai valita niitä. Valinnan jälkeen käyttäjä voi muiden toimintojen ohessa myös poistaa selvityksen.

Browse Surveys -sivulla voidaan selata kaikkia sovelluksen tietokannassa olevia selvityksiä (kuvio 11). Sivulla voidaan suoraa poistaa tai valita selvityksiä. Selvitys valitaan klikkaamalla selvitystä ja poistaminen tapahtuu x-painikkeesta. Jos selvitys valitaan, niin siirrytään suoraan sivulle, jolla selvityksiä myös luodaan.



KUVIO 11. Mobiilisovelluksen Browse Surveys -sivu.

4.4 Tietojen kerääminen mobiilisovelluksella

Mobiilisovelluksessa tietojen tallennus toteutettiin tehtäväksi automaattisesti samalla kun tietokenttiä täydennetään tai jokin kohta valitaan. Erillistä tallennus-painiketta ei ole käytössä. Tapa todettiin mobiililaitteille sopivaksi muihin sovelluksiin tutustumalla ja testauksella. Suurin osa täytettävistä kentistä on samantapaisia, joten muutaman tallennustapahtuman avulla voitiin tallentaa tiedot mobiilisovelluksen tietokantaan niiden muuttuessa.

4.5 Tietojen siirtäminen ja hakeminen mobiilisovelluksesta

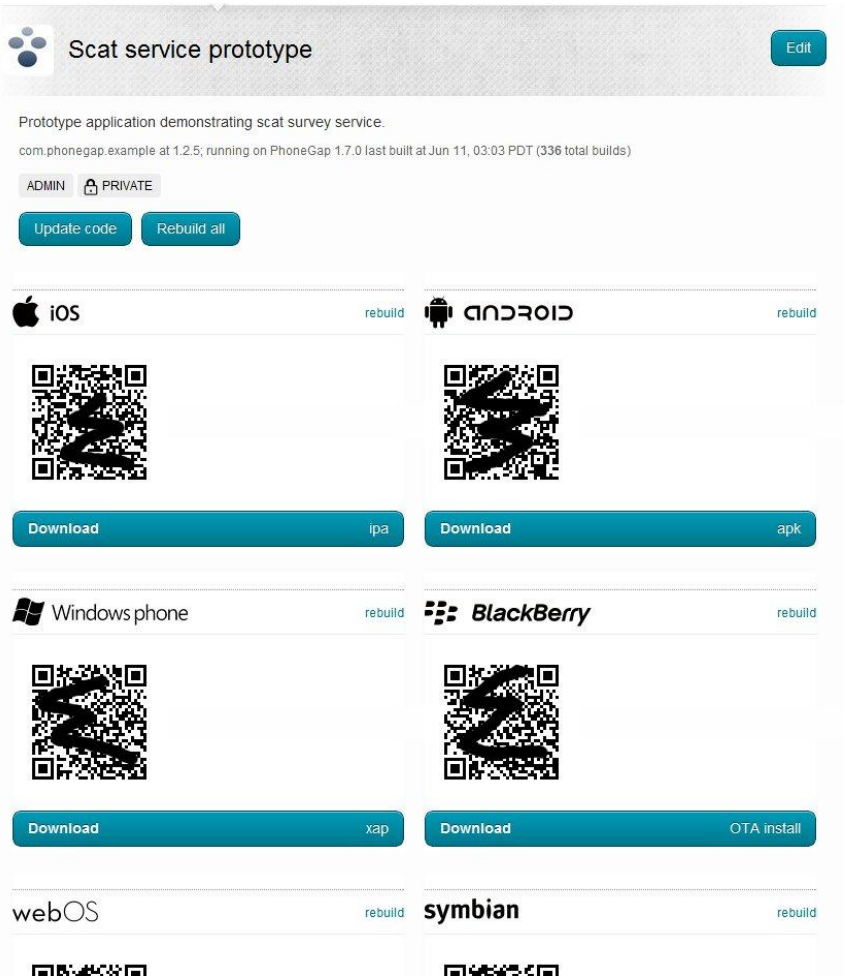
Tiedon siirto palvelimelle toteutettiin JavaScriptin POST-toiminnolla jQuery-kirjaston avulla. Siirrettävä tieto muutetaan base64-koodatuksi tekstiksi sisältäen salasanan ja käyttäjätunnuksen, jonka jälkeen se lähetetään palvelinsovellukseen, joka purkaa sen ja tallentaa tiedot tekstimuotona xml-tiedostoon. Tämä on tilapäinen ratkaisu prototyypissä. Täydellisessä järjestelmässä tiedot tallennettaisiin suoraan tietokantaan ja järjestettäisiin paremmin.

Prototyypissä palvelinsovelluksesta haettuja tietokantatietoja on vähän, sieltä haetaan ainoastaan testitietoina kaikki organisaatiot. Tämä tapahtuu sovelluksen käynnistyksen yhteydessä, jos Internet-yhteys on saatavilla. Haettu data, eli tässä tapauksessa organisaatiot on sisällettynä JSON-olioon, joka sovelluksen päivityksen yhteydessä tallentuu localStorageen, josta sitä voidaan käyttää suoraan jokaisella sivulla. Täydellisessä järjestelmässä tiedon haku palvelimelta pitää pystyä suorittamaan myös käyttäjän toimesta ja haettujen tietojen määrä olisi erittäin paljon suurempi.

4.6 Build.phonegap.com-pilvipalvelu

PhoneGap-pilvipalvelu oli olennainen osa prototyypin kehitystä. Sillä voitiin kääntää koodi automaattisesti kaikille PhoneGapin tukemille käyttöjärjestelmille pienine rajoituksineen. Sen avulla käännettiin lähes aina sovellus tablet-laitteelle. Poikkeuksena oli ensimmäinen Macintosh käännös, jolla käännettiin koodi iPad2:lle ja samalla luotiin pakollinen sertifikaatti PhoneGap-pilvipalvelua varten, jonka avulla voitiin myöhemmin kääntää iOS-koodia myös pilvipalvelussa.

Myös android-käännös tehtiin kerran testausmielessä Eclipsellä. Kuviossa 12 nähdään prototyypin sovellus pilvipalvelussa valmiina ladattavaksi.



KUVIO 12. Prototyyppi PhoneGapin pilvipalvelussa.

Pilvipalvelun käyttäminen on yksinkertaista sen jälkeen kun sinne on asetettu tarvittavat sertifikaatit (allekirjoitukset). Palvelussa on muutamia asetuksia sovellukseen liittyen, mutta on suositeltavaa käyttää config.xml-tiedostoa asetuksien määrittämiseen, koska se mahdollistaa myös sellaisten asetuksien määrittämisen, mitä itse sivustolla ei pysty tekemään, esimerkiksi sovelluksen kuvakkeet voidaan asettaa eri käyttöjärjestelmille.

Esimerkki config.xml-tiedostosta, joka oli käytössä prototyypissä:

```
<?xml version="1.0" encoding="UTF-8" ?>
   <widget xmlns = "http://www.w3.org/ns/widgets"
      xmlns:gap = "http://phonegap.com/ns/1.0"
      id        = "com.phonegap.example"
      versionCode="24"
      version   = "1.2.4">

   <name>Scat service prototype</name>

   <description>
      Prototype application demonstrating scat survey service.
   </description>

   <author href="http://yritys.fi/" email="henkilo.a@gmail.com">
      Henkilö A
      Yritys Oy
   </author>
<icon src="images/icons/ss10_67.png" />
</widget>
```

Tiedostossa määritellään sovelluksen perusasetuksia kuten versionumerot, nimi, kuvaus, tekijä ja ikoni. Tiedostossa voidaan myös määrittää halutun PhoneGap-version käyttö sekä muita asetuksia, esimerkiksi sallitut PhoneGap-ominaisuudet, joita voidaan käyttää, ja Internet-osoitteet, joista sovellus voi hakea tietoa.

4.7 Käyttäjän tunnistautuminen

Prototyypissä käyttäjän tunnistautuminen on keskeneräinen, mutta otettu huomioon. Käyttäjänä mobiilisovelluksessa käytetään testaustunnusta ja salasanaa. Palvelinsovelluksessa kaikki tunnukset ja salasanat hyväksytään. Täydellisessä järjestelmässä olevaa menetelmää ei ole vielä kokonaan suunniteltu, mutta mobiilisovelluksen kirjautuminen tapahtuu todennäköisesti palvelimella olevan tietokannan avulla.

5 TESTAUS

5.1 Prototyypin toimivuus

Prototyypin testaus suoritettiin päivittäin samalla kun prototyyppiä kehitettiin. Testauksessa haettiin parasta tapaa ladata sivuja siten, että mobiilisovellus toimisi mahdollisimman nopeasti ja vakaasti. Myös tietojen tallennuksen toimivuuden testaukseen perehdyttiin ja minimointiin tietojen häviäminen käymällä läpi kaikki mahdolliset tapahtumat, mitä mobiilisovelluksessa voisi esiintyä.

Tietojen häviäminen prototyypistä kuitenkin on mahdollista tietyssä päivitystilanteessa. Tärkeimmät tiedot ovat tallessa sisäisessä tietokannassa. Jos prototyyppi huomaa, että uusi päivitys on saatavilla, se aloittaa sen lataamisen ilman käyttäjältä kysymistä. Tämä voi pahimmillaan aiheuttaa sen, että uusi automaattisesti ladattava versio korvaa aiemman tietokannan jonkin muutoksen takia. Silloin kaikki tallennetut Survey-tiedot häviävät. Kysymällä käyttäjältä varmistuksen päivittämiseen, voitaisiin välttää tietojen katoamista.

5.2 Kenttätestaus

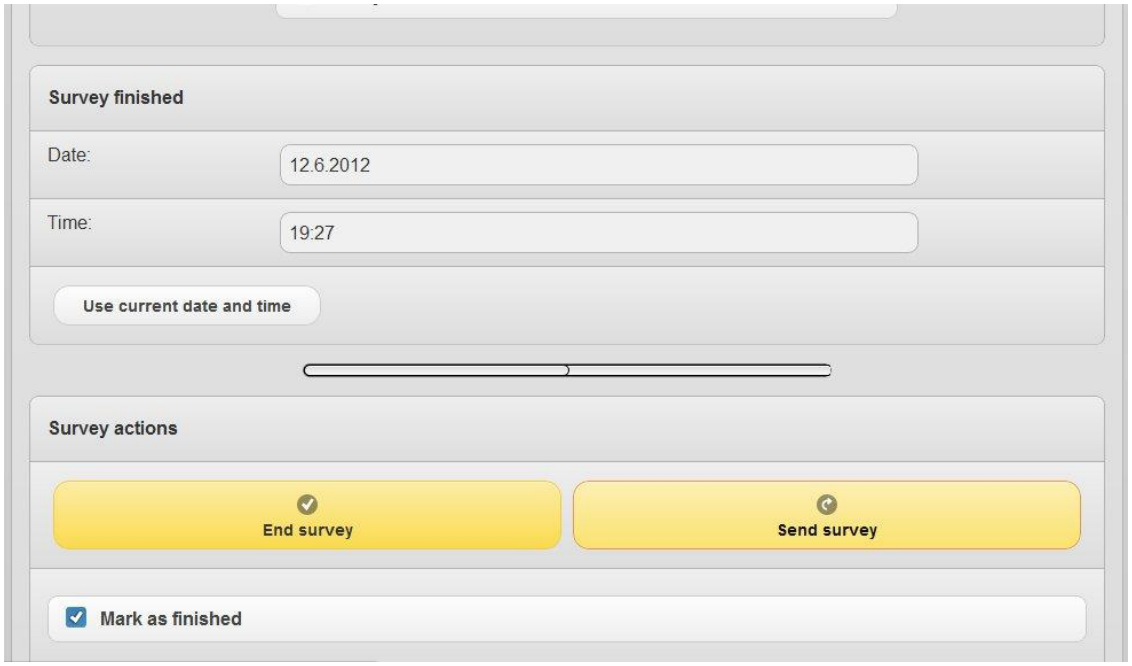
Mobiilisovelluksen täydellistä kenttätestausta ei päätetty suorittaa, koska toimivuus voitiin päätellä normaalitestauksen yhteydessä. Kuvia pystyi ottamaan ja tietojen tallennus toimi suurimmalta osalta. Prototyyppiin jäi pieni virheellinen toiminta, joka huomattiin testauksen yhteydessä. Yhdellä sivulla ei tallentuneet kaikki tekstitiedot. Korjaus päätettiin jättää jatkokehityksen yhteyteen.

GPS-tietojen tallentaminen toimi odotettua paremmin, koska tiedot saatiin usein nopeasti myös sisätiloissa. Joskus harvemmin kesti hetken aikaa ennen kuin laite onnistui hakemaan GPS-tiedot. Tämä saattoi tietyissä tilanteissa aiheuttaa virhetilanteen, mutta ei haitannut testausta.

Kerättyjen tietojen lähettäminen palvelimelle toimi suhteellisen hyvin. Itse tietojen lähettämien antoi virheilmoituksia testauksessa vain silloin, kun kuvien lähettämisessä oli jotain häiriötä. Tähän virheelliseen tilanteeseen päästiin esimerkiksi sillä tapaa, että luotiin sample ja siihen otettiin

kuva. Tämän jälkeen sample poistettiin ja kun tietoja yritettiin lähettää uudelleen, niin tapahtui virhetilanne. Virhe on melko yksinkertainen null-tarkistuksen puuttuminen.

Ulkoasuseikat toimivat niin kuin tarkoitettu lähes joka puolella sovellusta. Suurin ulkoasu virhe liittyy tietojen lähettämiseen palvelimelle ja siinä tarkemmin siihen miten tiedon siirron edistymispalkki näkyy silloin, kun tiedot lähetetään kaksi kertaa peräkkäin palvelimelle. Tällöin edistymispalkki ei resetoidu täysin oikein vaan se näkyy täysin harmaana (KUVIO 13), vaikka taustan pitäisi olla musta (KUVIO 14).



KUVIO 13. Virheellinen tiedonsiirron edistymispalkki.

KUVIO 14. Suunnitelmanmukainen tiedonsiirron edistymispalkki.

Testauksessa löydetyt virheet eivät juuri haitanneet mobiilisovelluksen käyttämistä prototyyppitarkoituksessa ja itse testausta voitiinkin pitää prototyypin onnistumisen merkkinä. Koska siinä näh-

tiin vaadittujen toimintojen toimivuus. Testaus alustoina käytettiin iPad 2 - ja Android-tablettia. Kummallakin tabletilla toimivuus oli lähes identtinen. Ainoat eroavaisuudet olivat erilainen suorituskyky ja pieni graafinen ero kuvien näyttämisessä. Android-tabletissa kuvat näkyivät liian isoina.

6 POHDINTA

Opinnäytetyön tavoitteena oli toteuttaa prototyyppi tiedonkeräysjärjestelmästä, jota käytettäisiin tablet-laitteilla. Tässä toimeksiantajan sekä oman mieleni mukaan onnistuttiin. Järjestelmästä luotiin helposti jatkokehitettävä täydelliseksi järjestelmäksi asti. Mahdollisesta jatkokehityksestä onkin puhuttu.

Opinnäytetyön edetessä tietosisältö muuttui jonkin verran. Olisi esimerkiksi ollut mahdollista, että olisi jouduttu kehittämään koko mobiilisovellus Macintoshilla, tällöin sisältö olisi ollut jotain ihan muuta. Tutkimuksen jälkeen päädyttiin siihen että käytetään PhoneGapiä, jotta kehitys tapahtuu samalla monelle eri alustalle ja voidaan käyttää ainakin Android-tabletteja sekä iPadejä, mutta myös muita. Alkuperäinen vaatimus oli pelkästään iPad 2:lle luotu sovellus.

Kun tutkimus, testaus ja uudet tavoitteet saatiin tutkittua, tehtyä ja mietittyä niin aloitin kehittämisen ASP.NET MVC3 -palvelinsovelluksesta. Siinä kehitys alkoi opettelemalla uutta MVC3-kehystä. Sen jälkeen toteutin perustietojen tallennus käsittelijän, jonka jälkeen lähdinkin toteuttamaan mobiilisovellusta. Mobiilisovelluksessa aluksi piti saada vain jotain tietoa siirrettyä testimielessä netin yli ASP.NET-palveluun. Tämän jälkeen pystyi jatkamaan kehitystä järkevimmillä tiedon tallennuksilla. Alkutestaus tapahtui ohjelmistokehityskoneella tavallisten web-sivujen avulla ilman mobiililaitetta. Tällä voitiin jo testata toimivuus suurimmalta osalta. Itse mobiilisovellus oli siis osa ASP.NET-sovellusta, josta se jälkeenpäin PhoneGapin tai Macintoshin avulla siirrettiin iPadiin ja/tai Android-tablettiin aina silloin tällöin.

Yksi vaatimus opinnäytetyön yhteydessä oli tehdä myös vaatimusmäärittely järjestelmästä. Sitä varten kävimme paljon yleisesi käytössä olevien SCAT-järjestelmien tietoja läpi toimeksiantajan kanssa ja keräsimme niistä tarvittavat toiminnot ja tietokannan rakenteen, mitä järjestelmässä tulee olla. Näitä kaikkia toimintoja ei siis tullut prototyyppiin, mutta ne otettiin huomioon ja sisällytettiin vaatimusmäärittelyyn.

Noin kuukausi siitä kun opinnäytetyön kehitys alkoi, niin käytössä oli vihdoin kaikki tarpeellinen eli Applen kehittäjän sertifikaatti, iPad 2 sekä Macintosh siihen, että tähän asti Macintoshilla tai web-selaimella testattu ”mobiilisovellus” voitiin testata itse iPad 2 -tabletissa. Ensi testaukset kuitenkin etenivät hyvin ja toimivuus oli lähes odotettua pienine ongelmineen, jotka nopeaa sitten korjaantuivat. Vähän myöhemmin käyttöön saatiin myös Android-tabletti, jossa toimivuus oli melkein samanlaista kuin iPad 2:ssa. Suurin ero oli laitteen teho, iPad 2 toimi paljon sulavammin kuin Android-tabletti, mutta myös pieniä ulkoasuun liittyviä seikkoja sisäisissä web-sivuissa oli. Ongelmat olivat lähinnä kuvien koissa, mutta ne saadaan korjattua kunnollisilla tyyliasetuksilla.

Tämän jälkeen kehitys mobiilisovelluksessa alkoikin edetä nopeammin, kun tiedettiin miten web-pohjaiset sivut toimivat PhoneGapllä itse tablet-laitteissa. Samalla sain määritettyä build.phonegap.com-sivuston käyttöön eli PhoneGap-pilvipalvelun. Siellä kääntäminen onnistui nopeasti kummallekin tabletille, joskin joskus palvelu kuitenkin hidasteli luultavasti ruuhkasta johtuen.

Opinnäytetyötä pidin kiinnostavana koko sen kehityksen ajan. Oli mukava opetella kokonaan uusia asioita kuten ASP.NET MVC 3, SQL Express, PhoneGap, localStorage, Web SQL Database, offline web pages, mobiilisovellukset, Xcode ja Macintosh. Opinnäytetyön laajuus oli sinänsä liian laaja, mutta sen pysty rajoittamaan sopivaksi. Itse opinnäytetyön tekeminen näin pitkälle ei olisi luultavasti onnistunut ilman aiempaa JavaScript sekä jQuery kokemusta, koska työ pääosin oli niillä kehittämistä ja niiden opetteluun työn aikana ei juuri tullut käytettyä aikaa. Noin puolet ohjelmointiajasta meni ASP.NET-palveluun, koska se vaati myös paljon opettelua. Siellä yksi hankalimmista asioista oli EXIF-tietojen eli kuvien sisältämien tietojen muokkaaminen. Erityisesti GPS-tietojen lisäämien luotuun kuvaan. Se tuotti hankaluuksia, koska esimerkkejä ei juuri löytynyt ja forumeillakaan ei osattu heti auttaa. Sen sai kuitenkin ajan kanssa toimimaan, kun yhdisteli eri puolelta löytynyttä tietoa. Kuvan tiedot piti sen takia lisätä kuvaan itse, koska PhoneGap:llä otetut kuvat eivät sisältäneet mitään EXIF-tietoja.

Opinnäytetyötä pidän onnistuneena ja hyvin opettavana, koska se keskittyi minulle aiemmin tuntemattomiin asioihin kuten tablet-laitteiden ohjelmointiin sekä MVC 3:n käyttöön. Opin paljon uusia asioita, joita voi käyttää jälkeenpäin hyväksi kaikenlaisessa ohjelmoinnissa.

LÄHTEET

Wikipedia. 2011a. Web SQL Database. Hakupäivä 30.11.2011, http://en.wikipedia.org/wiki/Web_SQL_Database.

W3C. 2010. Web SQL Database. Hakupäivä 11.6.2012, http://www.w3.org/TR/webdatabase/.

W3C. 2011. Web Storage. Hakupäivä 11.12.2011, http://dev.w3.org/html5/webstorage/.

HTML5ROCKS. Storage. Hakupäivä 11.12.2011, http://www.html5rocks.com/en/features/storage

Wikipedia. 2011b. JavaScript. Hakupäivä 11.12.2011, http://fi.wikipedia.org/wiki/JavaScript.

RFC 4329. 2012. Network Working Group. Hakupäivä 11.6.2012, http://www.apps.ietf.org/rfc/rfc4329.html#sec-7.2.

Korpela, J. 2009. Web-julkaisemisen opas. Hakupäivä 11.12.2011, http://www.cs.tut.fi/~jkorpela/webjulk/3.2.html.

PhoneGap. 2011. PhoneGap. Hakupäivä 12.12.2011, http://phonegap.com.

Wikipedia. 2011d. MVC-arkkitehtuuri. Hakupäivä 12.12.2011, http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri.

Wikipedia. 2011e. jQuery. Hakupäivä 14.12.2011, http://en.wikipedia.org/wiki/JQuery.

Wikipedia. 2011f. jQuery Mobile. Hakupäivä 14.12.2011, http://en.wikipedia.org/wiki/JQuery_Mobile.

Jquery Mobile. 2011. jQuery Mobile selects. Hakupäivä 14.12.2011, http://jquerymobile.com/demos/1.0/docs/forms/selects/.